Using RDF(S) to provide multiple views into a single ontology

Santtu Toivonen
Sonera Corp., Research
P.O. Box 145
FIN-00051 Sonera, Finland
(+358) 40 723 7217
santtu.toivonen@sonera.com

ABSTRACT

This paper deals with RDF (Resource Description Framework). The main point is to present a general model describing when and how to exploit RDF technology. It is suggested that RDF(S) [1] functions best as a means to provide mechanisms for expressing contextual and case-specific information. In other words, RDF(S) is suitable for providing different views into a single extensive ontology, rather than specifying the actual ontology. The ontology "behind" the case-specific RDF(S) is likely to be expressed using some other mechanism than RDF(S).

Keywords

Ontologies, concepts, domain-specificity, resources, properties, classes.

1. INTRODUCTION

1.1 Nature and scope of the paper

This paper is theoretical and methodological in its nature. It is theoretical since applications and implementation-specific details are excluded. It is methodological since it concentrates on the proper usage of RDF(S).

This paper introduces a simple model on how to exploit RDF(S) in large and heterogeneous environments that include several different applications. The opinion is that RDF(S) has a lot of useful features in describing resources, but also some drawbacks. After the model is presented, the possibilities as well as limitations of RDF(S) are discussed.

[1] RDF(S) refers to combined technologies of RDF and RDF-Schema. Cf. [19].

1.2 Technologies with significance to the model proposed here

RDF(S) technology aims at describing web resources. It is under development and standardization in the World Wide Web Consortium. RDF is specified in two separate documents, one about model and syntax of RDF [15] and the other about RDF schemas [4].

XML is one proposed representation format for RDF statements. Of the large amount of technologies in the XML family at least XML Namespaces [3] and XML Schema [1, 9 and 20] are relevant with respect to RDF. Namespaces are needed in RDF(S) because they help identifying the particular domains and modeling layers [19]. Furthermore, the particular RDF schema that is used for validating different RDF documents is identified using namespace notation. XML schema technology is needed for syntactic validation of RDF documents that are in XML format.

There are some differences between validation in XML and RDF [5]. Validation through RDF schemas grounds mainly on semantics, i.e. the meaning-based hierarchy and relations among the concepts to be defined. XML schemas perform syntactic validation instead; they concentrate on the grammar of the XML documents [7]. There is some semantics in XML schema technology, like the usage of datatypes, but compared with RDF schemas it is best thought of as a syntactic validation mechanism.

2. OVERVIEW OF THE MODEL

This chapter presents the general structure of the proposed model. The motivation is to familiarize the reader with different parts of the model.

Figure 1 presents the overview of the model and illustrates the role of RDF(S). Unlike in [5, 7, and 19], RDF(S) is not intended to cover the whole semantic categorization in the environment[2]. It is rather intended as a mechanism to provide domain-specific data related to small-scale tasks. An individual RDF document as

[2] Note that in [5, 7, and 19] standard RDF(S) is extended with a language called OIL (Ontology and Inference Language). Also DAML (DARPA Agent Markup Language) [14] extends RDF(S). What is proposed here, is different. Here the basic mechanisms of RDF(S) are thought to be such that RDF(S) (or any system with similar internal structure) is not suitable for describing a potentially large ontology.

well as an RDF schema document consists of a set of concepts that is likely to be subset of the concepts in the ontology.

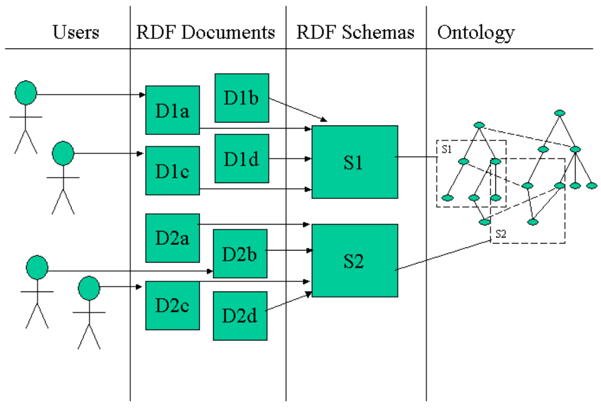

Figure 1. Overview of the proposed model.

Figure 1 is now examined from right to left. The rightmost section of the picture denotes ontology, the most general description of the environment in question. The next two sections are in the core focus of this paper. RDF schemas are seen as domain-specific validating filters. RDF documents are relatively small pieces of information that are validated against RDF schemas. And finally, users are the ones that utilize RDF(S) as their source of knowledge when working in the environment. Users can be software agents as well as human beings.

2.1 Ontology

Details of the ontology are outside the scope of this particular paper; ontology is treated here as a "black box". It could be implemented for example as a semantic network or a tree structure.

The approach in this paper favors the adoption of one big shared ontology as opposed to several smaller ones. It is acknowledged that this shared ontology might expand and become too slow and complicated to use. Additional limitations include the complexity and slowness of defining standards needed for one large heterogeneous ontology [6].

First one of the problems of the shared ontology approach, the slowness of using a large ontology, can be eliminated with RDF schemas. With RDF schemas it is possible to specialize the users of the ontology to be task-specific experts; they do not have to know every bit of information about the environment. The problems with defining standards for large ontologies are outside the scope of this paper.

The size and magnitude of the environment is a relevant question within the limits of this paper; how large and heterogeneous is the environment supposed to be? Instead of concentrating on domains, disciplines, business branches, etc., the concept of environment is used here along the following guideline: if two applications share one or more concepts, they belong in the same environment. And there should be only one ontology in one environment.

One important property of an ontology is extensibility [11]. It should be possible to introduce new concepts into an existing ontology so that the applications utilizing the ontology stay unbroken. In this model it is entirely possible to extend the ontology with new concepts since the users of the ontology operate using the RDF(S) that they themselves have defined. In the ontology all the concepts have similar ontological statuses[3]. RDF schemas and RDF documents together provide different views into the ontology.

2.2 RDF Schemas

RDF schemas are intended to function in a roughly similar role than DTD's function for XML documents. Individual RDF documents are validated against some RDF schema. RDF Schema specification [4] has defined a number of worthwhile concepts to be used when validating RDF documents. They are now presented briefly, since understanding their hierarchy and interrelations is important for the model presented in this paper.

At the topmost level the concepts are divided into three categories: `rdfs:Resource`, `rdfs:Class` and `rdf:Property`. Two important properties, `rdf:type` and `rdfs:subClassOf`, are needed in order to express the relationships among these concepts. Resource is the topmost class of the RDF system. Everything else is describable as a subclass of resource. Type-property is needed in order to express that each resource is a member of a class. A property is a specific aspect, characteristic, attribute, or relation used to describe a resource [15]. With respect to this paper, the division between properties and other resources is crucial[4].

The RDF schema specification [4] defines two important constraint properties: `rdfs:range` and `rdfs:domain`. These constraints are used only within RDF schemas; they do not appear in other RDF documents. The domain constraint indicates that a property may be used along with the resources of a certain class. For example, `author` is a property that could originate from a resource that is an instance of class `book`. A property may have zero, one, or more than one class as its domain.

Range, on the other hand, is something more rigorous; it specifies the class that the value of the property in question should be a resource of [4]. For example, a range constraint applying to the `author` property might express that the value of an `author` must be a resource of class `person`. A property can have at most one range property.

[3] This does not necessarily mean that the ontology is totally flat; there can naturally be some very general hierarchies among the concepts in the ontology. For example subclass - superclass - relation is something that can be said to hold between certain concepts regardless of the case-specific details.

[4] Every property is a resource and also a member of some class. In this paper properties are nevertheless often contrasted with classes and resources. The reason for this is to differentiate the concepts that get defined from those that participate in defining them.

2.3 RDF Documents

Individual RDF documents are validated against RDF schemas. RDF documents consist of descriptions, which in turn consist of statements. Each description represents some resource. Each statement represents some feature of the resource that is being described.

In RDF schemas the interrelations among selected resources and properties are defined. RDF documents contain naturally more specific and case-related data than RDF schemas; in RDF documents properties and resources are given values and thereby the ontological system is tied to actual instances of the resources. Following is a simple example of RDF in an XML syntax:

```
<RDF
xmlns="http://www.w3.org/1999/02/22-rdf-
       syntax-ns#">
  <Description
       about="http://www.santtusvideos.com/
       taxidriver.mpg">
    <director
     xmlns="http://www.santtusvideos.com/schema/">
       Martin Scorsese
    </director>
    <starring
     xmlns="http://www.santtusvideos.com/schema/">
       Robert De Niro
    </starring>
    <length
     xmlns="http://www.santtusvideos.com/schema/">
       114
    </length>
  </Description>
</RDF>
```

Here the mpg-version of *Taxi Driver* is presented as an RDF resource. It has three properties: director, starring and length. These are specified in individual statements. This document is validated against the RDF schema of an imaginary web video service called Santtu's Videos. Here all the statements refer to the same schema but this is not necessary. Features of a given resource could be defined in separate schema documents.

The shared ontology approach is favored in this paper over the multiple ontologies approach [6]. Nonetheless the usage of RDF(S) adopts some features from the multiple ontologies point of view. RDF documents and schemas are often organized into a hierarchy and descriptions might specialize other descriptions defined in other RDF(S)'s using `rdfs:subClassOf` and `rdfs:subPropertyOf` properties. It is good to keep in mind, however, that RDF(S) is treated as a means to provide views into ontologies, rather than specifying the actual ontologies.

2.4 Users

RDF(S) is intended to provide metadata about web resources that is both human-readable and machine-understandable [15]. Hence the users of the model proposed here can consist of software agents in addition to human beings. For example in FIPA (Foundation for Intelligent Physical Agents) there is already work done around the usage of RDF as a content language for software agents [10].

The semantic information that software agents use should be mainly external to the agents themselves [21]. Agents should have access to an external ontology describing the general structure of the environment. The ontology would constitute an independent repository of information. This way the agents themselves would not become "walking encyclopaedias" (cf. [8]) but remain relatively simple.

This semantic information external to the agents is distributed to all the other parts of the proposed model: RDF documents, RDF schemas, and the ontology. The agents should be designed so that they understand one or more RDF schemas. The agents can utilize individual RDF documents as pieces of case-specific information. They can do different things with the documents (and applications/services bound to the documents) according to their internal inference rules and the RDF schema/schemas they are committed to.

Also human users of the environment benefit from this model. The model helps people to understand different parts of the environment and applications appearing in it. For example, if someone decides to introduce a new service or resource into the environment, he can examine the schemas of the existing resources (assuming that the schemas are publicly available for examination).

If he finds a suitable schema, he can utilize it as a means to exploit the ontology. If no schemas as such work for the developer, there still might be some guidelines or pieces of information in existing schemas that help the developer to get started. Either way, people working in an environment with shared ontology can reduce the amount of work with public RDF schemas and this way avoid re-inventing the wheel.

3. USAGE OF RDF(S)

3.1 RDF and web resources

Again: RDF is intended to provide metadata about web resources. Different web resources naturally have different means of categorization. For example, libraries, video stores and digital phone books use different concepts as metadata [2]. There are nevertheless some common aspects among all of these. First, they all have resources, be they books, movies or phone book entries. Second, they all have properties characterizing the resources. Movies, for example, have actors, directors, length of the movie, etc. Third, the resources may be grouped into classes. There might be a class called movies and it might have a subclass called horror movies.

3.2 Properties in RDF(S)

RDF(S) is in this paper proposed as a means to provide case-specific information rather than means to constitute the whole ontology. The reason for this reduces to the question concerning the ontological status of properties. In [15] properties are described the following way: "A property is a specific aspect, characteristic, attribute, or relation used to describe a resource".

RDF(S) properties are hereby qualifiers that characterize some resources. They have a clearly different ontological status than classes, for example. Classes are something that are defined (definienda, sing. definiendum), properties are something that participate in defining them (definientia, sing. definiens). This is fully acceptable as long as the case-specificity and contextuality of the model is kept in mind. Depending on the context, concept c1 can be an attribute of c2 and vice versa [17].

In other words: Concepts that are definienda in some case or application form the basic level of concepts for that particular case. In RDF(S) terminology these would be classes to be defined. There are two other levels in addition: subordinate and superordinate level. Definiens (property in RDF(S) terminology) is at the subordinate level when compared with definiendum. Depending on the domain, however, relations between these levels vary (cf. [18 and 16]).

3.3 An example characterizing the case-specificity of RDF(S)

From the electronic video store's point of view director is a property that characterizes the resources of a class called movie. This is fine as long as it is clear that somewhere else director could appear also as a class that gets defined by some other properties. An electronic catalog of artists might have director as a class[5]. Now directors could have movies that they have directed as their properties. Just the other way around than in the video store[6]. This is illustrated in Figure 2.

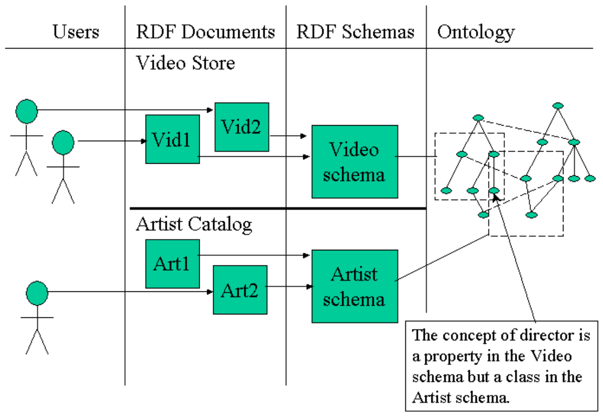

Figure 2. An example about the domain-specificity.

When representing the world, the structure of concepts should be as analogous to reality as possible [17]. And there is no a priori way to declare that some concept functions always as definiens while some other is always definiendum. That is why concepts should not be universally placed in either of these categories. In the end all concepts are similar with respect to their ontological statuses. RDF(S) is a technology with no good conventions that help coping with this matter.

Of course it is possible to introduce all (or at least majority of) concepts twice; once with the status of definiendum and again with the status of definiens. However, this is not a desirable solution. It leads to compatibility problems and violates the simplicity principle of ontologies. The principle states that there should be as few ontological commitments in an ontology as possible [11]. Introducing all concepts twice would cause a situation found in Figure 3.

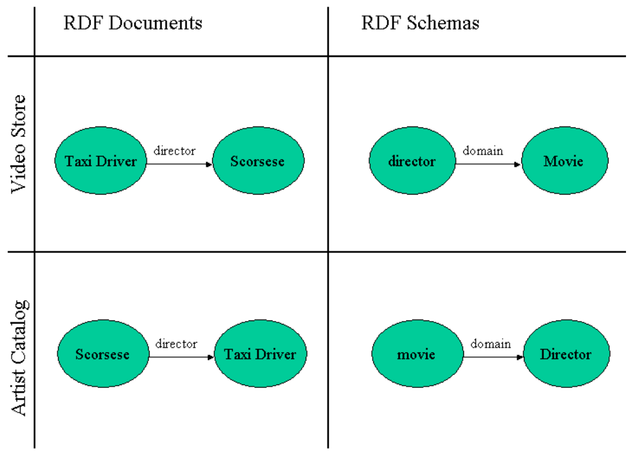

Figure 3. Usage of concepts in different cases.

First the RDF documents column of the Figure 3 is examined. It tells us that in video store the movie *Taxi Driver* has a property named director with the value "Scorsese". Artist catalog, on the other hand, has the director Martin Scorsese with a property named director that has the value "Taxi Driver". So *Taxi Driver* and Scorsese both appear once as resources to be defined and once as properties. The same thing concerns the RDF schemas column of the picture. In video store director is a property that belongs in the domain[7] of movies. In artist catalog the situation is contrary.

[5] Artist catalog would probably have artist as a basic level concept and director as subordinate level concept. However, in RDF(S) terminology these would both be classes, not properties.

[6] In principle even two different video stores could interpret the hierarchy of some set of concepts variously.

[7] For the sake of simplicity only one constraint property is presented here. Besides domain, also range is a useful property to be exploited in RDF schemas. In the schema of video store, for example, there could be a range constraint property named person attached to the director property. This would mean that the value of the director property is always a member of the class person. Furthermore, only one domain for each property is presented. If necessary, though, director could have other domains besides movies. It could be attached to TV-series, theatre plays, etc.

A phenomenon closely related to this is observed in [13]; different RDF schemas can specialize some class defined in another schema with `rdfs:subClassOf` and `rdfs:subPropertyOf` properties. They can use the same name but different definitions for that class in their own specializations. So there could be an upper RDF schema that has movies, directors, etc. all as classes. However, when electronic video store and electronic artist catalog specialize the classes in their own unique ways, the system as a whole becomes incoherent. This is one of the basic drawbacks of the multiple ontologies approach.

One remark here could be that since RDF is intended for describing web resources, the movie *Taxi Driver* (at least in mpg-format as in the code example presented earlier) is more appropriate candidate for a web resource than the director Martin Scorsese. That is because Martin Scorsese can not appear in a format distributed in the Internet unlike *Taxi Driver*.

Ontologically speaking, however, the movie *Taxi Driver* is not the same entity as the mpg-version of it distributed in the net. It is rather an abstract thing that has different instances. Compared with object-oriented programming, the movie *Taxi Driver* would be a class and the copies of that movie (for example the mpg-version distributed in the electronic video store) in turn instances of the class.

4. CONCLUSION

The expressive power of RDF(S) does not necessarily complete all the parts that are needed for expressing a semantic description of some system[8]. An ontology independent of the domain-specific details of its usage is needed. There should be an "isolated basic backbone" of ontology that is independent of any case-specific details [12]. And based on the arguments and examples presented here, it should be clear that RDF(S) alone does not fit together with this requirement.

What RDF(S) technology can do, however, is to provide means to access an ontology characterizing some environment – no matter how large or heterogeneous – in many ways.

5. DISCUSSION

5.1 Rethinking the properties

The main problem of RDF(S) presented in this paper is the division of the concepts into properties and classes. The answer proposed to this problem is the usage of an external ontology in addition to the RDF(S). From the ontology's point of view the usage of concepts in different RDF(S) is based on roles [12]; `rdfs:Resource`, `rdfs:Class` and `rdf:Property` are different roles of some concept defined in the ontology.

Another attempt to resolve this would be to reformulate the properties. Earlier an example of using director as a property belonging in the domain of movie in one place and movie as a property belonging in the domain of director in another was presented. Why not use directors and movies always as classes? Movie would have a property directed_by that would have a director as its value. Director would have has_directed property that would have a movie as its value.

At first sight this might seem wise. More carefully examined, however, this leads to a situation not preferable to defining every concept twice; once as a property and once as a class. The number of properties would be doubled as shown in figure 4; has_directed and directed_by would both exist between a movie and its director even though they have the same information content. This would again violate the simplicity principle of ontologies [11].

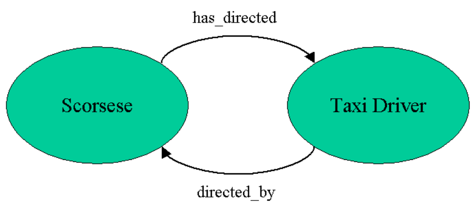

Figure 4. Duplicating the properties

Yet another attempt to overcome the problem of classes versus properties is reacting to it at levels residing on top of RDF. OIL (see [5, 7, and 19]) and DAML (see [14]) are examples of languages that are on a higher level than RDF.

However, introducing rules and restrictions that cope with limitations of RDF at a higher level does not seem feasible. For one thing, this again violates the simplicity principle of ontologies [11]; for each ambiguous class-property distinction at the RDF level there would exist a fixing principle at a higher level. Secondly, the whole idea of coping with problems of some level at another is not desirable; each level should be clear enough not to require fixing or configuring at other levels.

5.2 Deducing the ontology from RDF(S)

Here the ontology "behind" the RDF(S) is treated as a "black box". Its detailed structure is not discussed. It could however be possible (even in the model proposed in this paper) that the whole ontology is deducable from the total amount of RDF documents and schemas in a given environment. This depends on the interpretation of domain-specificity.

If all the concepts and their interrelations are such that they are found in the ontology it could be possible to make that deduction. Possible conflicts should however try to be avoided. If some concept is a property (definiens) in one schema and a resource to be defined (definiendum) in another, does that have any impact on the ontology? If it does, which one of the schemas determines the "ontological" location of the concept in question. If it does not, how it is possible to construct any hierarchy in the ontology (since there would be nothing in addition to the schemas)? On the other hand, if there are some general relations or attributes at the ontological level that are not visible in RDF(S), the deduction is not possible.

[8] See nevertheless "Discussion" for possibilities to deduce the ontology from RDF(S).

Clearly a deduction in the other direction is not possible. There is no way of knowing how the concepts in the ontologies are used and grouped in different RDF(S). This means that it is not possible to deduce all imaginable RDF(S) just by examining the ontology. And this is due to the proposed case-specific nature of RDF(S).

Acknowledgements. Thanks to Joose Niemistö and Johannes Gröhn for their help on this article.

6. REFERENCES

[1] Biron, P.V. and Malhotra, A. XML Schema Part 2: Datatypes. Technical Report, W3C, 2001. W3C Proposed Recommendation. http://www.w3.org/TR/xmlschema-2/.

[2] Bray, T. RDF and Metadata. 1998. http://www.xml.com/xml/pub/98/06/rdf.html.

[3] Bray, T., Hollander, D., and Layman, A. Namespaces in XML. Technical report, W3C, 1999. W3C Recommendation. http://www.w3.org/TR/REC-xml-names.

[4] Brickley, D., and Guha, R.V. Resource description framework (RDF) schema specification. Technical report, W3C, 2000. W3C Candidate Recommendation. http://www.w3.org/TR/rdf-schema.

[5] Broekstra, J., Klein, M., Decker, S., Fensel, D., van Harmelen, F., and Horrocks, I. Enabling knowledge representation on the web by extending RDF Schema. 2000.

[6] Cui, Z., Tamma, V. and Bellifemine, F. Ontology Management in Enterprises, British Telecommunications Technology Journal, October, 1999.

[7] Decker, S., van Harmelen, F., Broekstra, J., Erdmann, M., Fensel, D., Horrocks, I., Klein, M., and Melnik, S. The Semantic Web - on the Roles of XML and RDF. In IEEE Internet Computing. September/October 2000.

[8] Dennett, D.C. When Philosophers Encounter Artificial Intelligence. In Daedalus, Proceedings of the American Academy of Arts and Sciences, 117, 283-295, 1988.

[9] Fallside, D.C. XML Schema Part 0: Primer. Technical Report, W3C, 2001. W3C Proposed Recommendation. http://www.w3.org/TR/xmlschema-0/.

[10] FIPA RDF Content Language Specification. 2000. http://www.fipa.org/specs/fipa00011/XC00011A.pdf.

[11] Gruber, T.R. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In Nicola Guarino & Roberto Poli (eds.): Formal Ontology in Conceptual Analysis and Knowledge Representation. Padova, Italy: Kluwer Academic Publishers, 1993.

[12] Guarino, N. Some Ontological Principles for Designing Upper Level Lexical Resources. Proceedings of First International Conference on Language Resources and Evaluation, 527-534, Granada, Spain, 1998. ELRA - European Language Resources Association.

[13] Heflin, J., and Hendler, J. Semantic Interoperability on the Web. In Proceedings of Extreme Markup Languages 2000.

[14] Hendler, J. and McGuinness, D.L. The DARPA Agent Markup Language. In IEEE Intelligent Systems, Vol. 15, No. 6, November/December 2000, 67-73.

[15] Lassila, O., and Swick, R.R. Resource description framework (RDF) model and syntax specification. Technical report, W3C, 1999. W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax.

[16] Murphy, G.L., and Lassaline, M.E. Hierarchical Structure in Concepts and the Basic Level of Categorization. In Koen Lamberts and David Shanks (eds.): Knowledge, Concepts, and Categories, 93-131. Hove: Psychology Press, 1997.

[17] Saariluoma, P. Foundational analysis: Presuppositions in experimental psychology. London: Routledge, 1997.

[18] Saeed, J.I. Semantics. Oxford: Blackwell Publishers Ltd, 1997.

[19] Staab, S., Erdmann, M., Maedche, A., and Decker, S. An Extensible Approach for Modeling Ontologies in RDF(S). In First Workshop on the Semantic Web at the Fourth European Conference on Digital Libraries, Lisbon, Portug*al*, 2000.

[20] Thompson, H.S., Beech, D., Maloney, M., and Mendelsohn, M. XML Schema Part 1: Structure. Technical Report, W3C, 2001. W3C Proposed Recommendation. http://www.w3.org/TR/xmlschema-1/.

[21] Toivonen, S. Definition and usage of a software agent. In Arpakannus (2/2000), 9-13, 2000.